- **Overview**
  - This framework will provide additional tooling to DTP for data viewing and mining including basic formatting, filtering, searching, and sorting. The framework will be extendable to application specific domains. It is expected that application developers will create domain specific actions that will operate on selected data as rows, columns, or cells.
- **Data Display**
  - A user may be accessing a database with many columns. The user may only be interested in a subset of those columns. For performace, it would be best to store a row as an array. While viewing the data, the user may find that an additional column is required. Adding a new column to a static array will have poor performance. A query should specify the columns retrieved from the database which will be stored as an array for each row. A user may show or hide any column within the data retrieved. If the user required a column that was not retrieved, a new query is made and a new table is populated. It will be up to the user to decide the trade-off between memory and functionality. The user might query 10 columns, but only display 5 thus burning memory for an additional 5 columns with the trade-off of allowing the other 5 to be displayed quickly.
  - Background data population
    - Need to explore the ability to populate columns in the background. The bulk of the table data would be displayed, but some columns would be empy until their data is retrieved in the background.
  - Editing
    - Deferred updates
      - Updates happen as a background task - this may require application specific server support.
    - Change multiple fields
      - Change one or more fields from the selected rows with a single dialog. The value of a specified field in the dialog is applied to all selected rows.
  - Display types
    - Table
    - Tree
  - Format
    - Format information will be stored in an EMF model as part of the query
    - Column
      - Visibility
        - Specifies which columns are visible
      - Order
        - Specifies the order in which the columns will be displayed
      - Sorting
        - Specifies the columns and their priorities for sorting the data
      - Movable
        - Specifies which columns are movable
      - Justification
        - Specifies the column justification (left, center, right)
    - Data
      - Format
        - Data formatters may be provided for specific data types. For example a Date may be displayed in several different formats
      - Font
        - Fonts may be applied to data statically or algorithmically
      - Color
        - Foreground and background colors may be applied to data statically or algorithmically
      - Grouping
        - Rows may be grouped together based on a common column. This will turn the table into a tree

with columns. The grouping should be specified in the format section of the query.

- Actions
  - A generic set of actions will be provided. Domain specific actions may be provided by extensions. Actions performed on a collapsed group apply the action to the entire group.
  - Create query / filter
    - A new query / filter may be created based on column or row selections
  - Column
    - Move
    - Hide
    - Show
    - Sort
    - Format
  - Row
    - Hide selected / not selected
      - Hiding rows will push the set of hidden rows on a stack
    - Mark selected / not selected
      - Changes the row color or uses a checkbox
    - Invert selection
    - Show
      - Display the last set of hidden rows poping them from the stack. When the stack is empty, show will be disabled
    - Show all
      - Displays all hidden rows
  - Searching
    - Optional case-sensitivity
    - Select / mark found items / rows
    - regex match, string match, color match
  - Sorting
    - Left click on a column will toggle column sorting ascending vs descending - this will override column sorting specified in the query
  - Marking
    - Algorithmic marking (mark all rows where the date cell is in the future)
    - Marking could be entire row or a single cell
    - Use colors, fonts, and decorators
- Computed columns
  - Computed at query time
  - Lazy (background) computation - column loads wile being viewed
  - Hooks for plugging in computation modules (extension point)
  - Default support for sum, avg, min, max, etc
- User Actions
  - Add buttons that launch scripts
  - Need a mechanism for a running script to provide feedback / error status
- Refresh
  - Automatically refresh the data controllable via preference
  - Manually refresh the data

- **Data Query**
  - Need the ability for domain experts to specify a default query
  - Source selection
    - Data may be stored in a database, flat file, or any other permanent store.
    - Editor should hide database structure
    - MultipleDataSources

- Ability to discover data sources
- Combine multiple data sources on a single view
- Mapping from a column in one data source to a column in an "unrelated" data source (create a foreign key on-the-fly)

- Column selection & formatting
  - Joins should be transparent to the user
  - Data model
- Filter

- **Data Context**
  - Since different application domains may co-exist in the same deployment of a tool that uses DTP, a context may be necessary to properly populate the context menus. For example, it doesn't make sense to see actions for running junit tests if you are viewing stock market data.
- **Query Templates**
  - Domain specific templates may be provided to simplify the query editing process. These templates could provide a list of data sources, the columns that are typically used, and even a starter filter.
- **Notes**
  - Bugzilla 23103 must be fixed to support this work